МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«ИЗМЕРЕНИЕ СТЕПЕНИ АССОЦИАТИВНОСТИ КЭШ-ПАМЯТИ»

студента 2 курса, 23202 группы

Пятанова Михаила Юрьевича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Кандидат технических наук
Владислав Александрович
Перепёлкин

Новосибирск 2024

СОДЕРЖАНИЕ

ЦЕЛЬ

1. Экспериментальное определение степени ассоциативности кэш-памяти

ОПИСАНИЕ РАБОТЫ

В ходе выполнения лабораторной работы были выбраны следующие параметры для теста:

1. Размер фрагментов: 32 КБ., что соответствует размеру L1d кэша.
2. Величина смещения: 16 МБ.

Был написан алгоритм (прил. 1), заполняющий массив и измеряющий среднее время доступа к элементу массива.

Графики зависимости среднего времени доступа к элементу массива от числа фрагментов представлены в приложении 3. Они были нарисованы благодаря языку программирования Python, листинг представлен во втором приложении.

На основе графика для -O1 можно сделать такой вывод:

1. Степень ассоциативности L1 и L2 уровня кэш-памяти: 8
2. Степень ассоциативности L3 уровня кэш-памяти: 16

Что соответствует реальным значениям.

ЗАКЛЮЧЕНИЕ

Было проведено экспериментальное определение степени ассоциативности кэш-памяти. Были получены следующие результаты:

1. Степень ассоциативности L1 и L2 уровня кэш-памяти: 8
2. Степень ассоциативности L3 уровня кэш-памяти: 16

Приложение 1.

Команда для компиляции:
g++ lab9.cpp -o ex
g++ lab9.cpp -o ex -O1

```cpp
#include <fstream>

uint64_t getCpuTicks() {
    unsigned int lo, hi;
    __asm__ __volatile__ ("rdtsc" : "=a" (lo), "=d" (hi));
    return ((unsigned long long)hi << 32) | lo;
}

int* getArray(size_t n, size_t size, size_t offset) {
    size_t arraySize = offset * n;
    int* arr = new int[arraySize];
    for (size_t i = 0; i < size / n; ++i) {
        for (size_t j = 0; j < n - 1; ++j) {
            arr[i + j * offset] = (i + j * offset + offset);
        }
        arr[i + (n - 1) * offset] = i + 1;
    }
    arr[size / n - 1 + (n - 1) * offset] = 0;
    return arr;
}

uint64_t measure(int* array, size_t size) {
    volatile size_t k, i;
    uint64_t start = getCpuTicks();
    for (k = 0, i = 0; i < size; i++) {
        k = array[k];
    }
    uint64_t end = getCpuTicks();
    return (end - start) / size;
}

int main() {
    // 32KB L1d.
    const size_t size = (32 * 1024) / sizeof(size_t);
    // 16MB Offset.
    const size_t offset = (16 * 1024 * 1024) / sizeof(size_t);
    // Maximum number of the fragments.
    const size_t N = 32;
    // Iterations to get the most precise result.
```

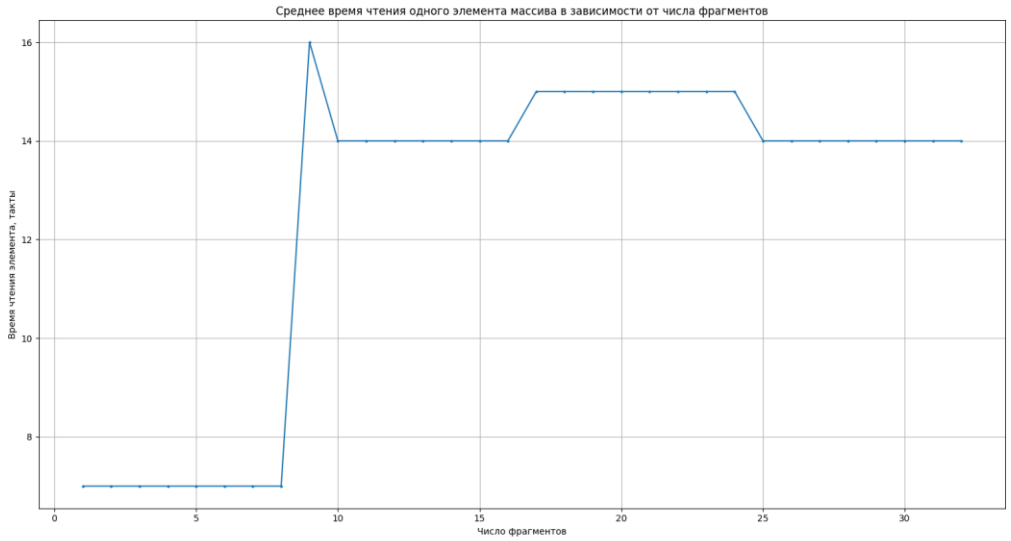
```
    const size_t iterations = 1000;

    std::ofstream out("lab9.txt");

    for (size_t n = 1; n < N + 1; ++n) {
        int* arr = getArray(n, size, offset);

        uint64_t minTicks = UINT64_MAX;
        for (size_t i = 0; i < iterations; ++i) {
            uint64_t ticks = measure(arr, size);
            minTicks = minTicks > ticks ? ticks : minTicks;
        }
        out << n << ' ' << minTicks << '\n';

        delete[](arr);
    }
    out.close();

    return 0;
}
```

Приложение 2.

```
import matplotlib.pyplot as plt
import numpy as np

def read_data(file_path):
    x_values = []
    y_values = []
    with open(file_path, 'r') as file:
        for line in file:
            x, y = map(float, line.strip().split())
            x_values.append(x)
            y_values.append(y)
    return x_values, y_values

def plot_data(x_values, y_values):
    plt.plot(x_values, y_values, marker='o', linestyle='-',
markersize=2)
    plt.xlabel('Число фрагментов')
    plt.ylabel('Время чтения элемента, такты')
    plt.title('Среднее время чтения одного элемента массива в
зависимости от числа фрагментов')
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    file_path = 'lab9.txt'
    x_values, y_values = read_data(file_path)
    plot_data(x_values, y_values)
```

Приложение 3.

Без флага оптимизации -O1:



С флагом: